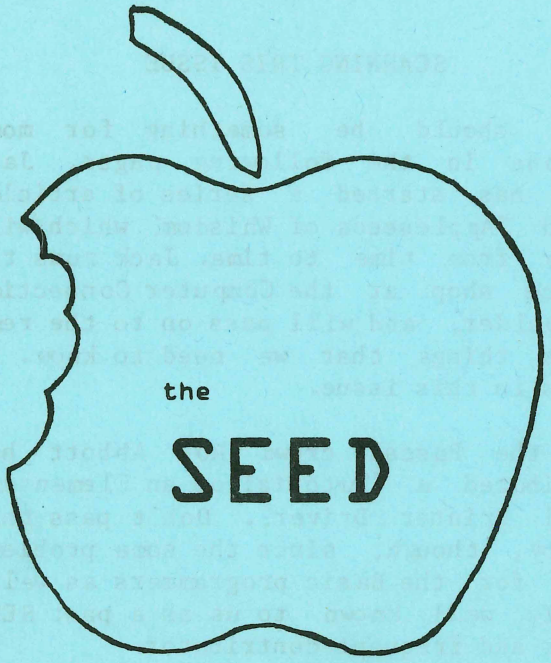

OFFICERS

PRESIDENT
Steve Martin (303) 469-6855
VICE PRESIDENT
Dave Eisler (303)694-6217
TREASURER
Fred Yates (303)988-9053
SECRETARY
Dave Moger --
LIBRARIAN
Bob Gaiser (303)779-4033
SEED EDITORS
Frank Reeves (303)499-2192
Louis Breyfogle (303)449-6454
Ron Abbott (303)772-7198
OFFICERS-AT-LARGE
Dan Geisler (303)832-6465
Al Tara (303)755-8009

Volume 4 Number 4 August 1982 Price $1.00

MEMBERSHIP INFORMATION
$10 APPLICATION FEE AND $18 PER YEAR DUES

Seed Editors
3015A Bluff St.
Boulder, CO 80301

FIRST CLASS MAIL
U.S. POSTAGE
PAID
Permit No. 502
Boulder, CO 80302

ADDRESS CORRECTION REQUESTED

PRINTED BY KWIK KOPY

BYTES FROM THE EDITORS

"Hey Ma, look! There really is a SEED!" After a rather long spell, three members have come forward to act as co-editors of the SEED. You will find our names on the masthead. Why three editors? It's simply too much for one individual to do and do well. With three of us, it's hoped that two can cover for the third when things get tight.

This issue of the SEED is not intended to represent what your editors want it to be, but we feel that it is a good start. As time goes on, and you let us know your wants, we hope to improve the finished product. Please remember that APPLE PI is made up of a very diverse group of people whose interests and experience vary quite widely. Therefore, your editors request that you feel free to submit anything that you wish to share with the membership to the editors for possible publication. If you have an idea for an article but don't feel that you are capable of writing it, let one of the editors know, and we will try and find someone to write it.

Your experience with hardware or software for the APPLE can be invaluable to other members. If you have a new goodie, write a review and let others know what you think of it, good or bad. Found a new way of doing something? Write and article and let the rest of us in on it too.

Above all, let us know what you think of what we are doing. After all, it's YOUR SEED. We need and want YOUR contributions and reactions.

WANTED!

STAFF CARTOONIST

Do you have the ability to draw humorous cartoons? Do you like to editorialize with pen or brush about computers and the people who make them go? If so, we want YOU! The pay is lousy but the recognition is grand. Contact one of the editors if you would like the job.

SCANNING THIS ISSUE

There should be something for most everyone in the following pages. Jack Quinn has started a series of articles called 'Appleseeds of Whisdom' which will appear from time to time. Jack runs the service shop at the Computer Connection in Boulder, and will pass on to the rest of us things that we need to know. It starts in this issue.

For the Pascal crowd Ron Abbott has contributed a 'Tutorial on an Elementary Pascal Printer Driver'. Don't pass this one by, though, since the same problems exist for the Basic programmers as well. Ron is well known to us as a past SEED editor and frequent contributor.

Lou Breyfogle has two brief ones, a product review of the HP-16C calculator geared to the computerist, and a table that lets us translate 3.3DOS RAM addresses to their proper track and sector locations on a disk. ENJOY!

LDB

ARTICLE FORMAT

For the moment, we would appreciate your submitting your articles in machine readable form. This implies that you give us a disk with one of the following on it:

3.3 DOS TEXT FILE (Apple Writer II)
3.3 DOS BINARY FILE (Apple Writer I)
APPLE PASCAL EDITOR FILE

Your disk will be returned to you promptly. Be sure that you give us a lable telling what you are giving us.

We are printing this issue with a Diablo using a Prestige Elite 12 pitch daisywheel, a column width of 3.5 inches (42 characters), and 56 lines per column. It is our intent to print program listings in 80 character widths. Diagrams and sketches will be delt with on an individual basis - they do not necessarily have to be furnished camera ready, but we do need to be able to understand them!

The HP16C - Product Review

by Louis D. Breyfogle

A fellow APPLE owner stopped by a few days ago, handed me a small black case, saying "I'll bet you have your own before I come back for it", and left. He was right!

The Hewlett-Packard HP16C is one of a line of 3 X 5 inch pocket calculators that can make life a lot easier for the computer hacker, especially if he is trying to learn assembly language programming. It is aptly named "The Computer Scientist" calculator.

A first glance at the HP16C shows 39 keys, most of which are actually capable of three different functions. Readout is on a ten digit liquid crystal display. Since the calculator actually does its work in binary (just like the APPLE) it is possible to examine its binary words directly up to a 64 bit word length by moving its 10 bit "window" along the word with keyboard control. It will display the results in HEX, Decimal and Octal as well, converting between bases with a single keystroke. This operation is done in integer math, but two keystrokes will convert the calculator to floating point and/or scientific notation if that is appropriate for what you want to do. Operation in 1's complement or 2's complement mode is keystroke selectable.

The HP16C is keystroke programmable to a maximum of 203 program lines, allowing sufficient flexability for the program to make decisions and perform iterations through conditional and unconditional branching. Since keyboard entry is in RPN (Reverse Polish Notation) it is unnecessary to use parentheses in the program. There is a four level stack register, a LastX register and up to 32 storage registers that can be addressed directly by name from either the keyboard or a program. GOSUBs and GOTOs are also available to use.

The logical (Boolean) operations NOT, OR, AND, and EXCLUSIVE OR return the results of a bit-by-bit analysis of a binary number. Two types of shifts are supported, a logical shift or an arithmetic shift, shifting left or right as necessary. A carry flag is activated when appropriate. In case you have managed to produce a result in the calculator that is outside its range, large or small, an "Out of Range" flag is activated. Bit masking is also supported by the HP16C.

The use of a liquid crystal display and low power circuitry results in an extremely long battery life. As a result, no battery charger is available. A set of three fresh silver-oxide batteries will last at least 180 hours of continuous program running. With the display turned off, the batteries will preserve the contents of Continuous Memory for at least two years. The Continuous Memory will even survive without batteries long enough to install a new set! A flashing asterisk on the display warns you when the batteries need to be replaced. Its quite remarkable to turn the display off and come back the next day and find your programs intact, ready to start running where you left off.

The calculator comes with a soft case and will easily fit into a shirt pocket. The HP16C comes with a comprehensive 133 page Owners Handbook that is not much bigger than the calculator, but is extremely well written.

The HP16C has a retail price of $150.00, but is being discounted by at least one dealer in the Denver area to $132.00. After I had bought mine, I learned of one mail order source that was discounting it to $117.95.

Is it worth this kind of money? Like anything else, its utility value is as different as the people who know about it. If you are in the process of trying to learn assembly language programming go look at one. The ability to key in subroutines and observe the results in binary, hex and decimal makes it much easier to grasp exactly what is happening. Bit manipulations, such as in the VTOC of a disk, become trivial with the HP16C.

APPLESEEDS OF WISDOM

Jack D. Quinn

It's a nice quiet evening (the kids are in bed) and you have just settled down in front of your Apple to play the latest game that you just picked up from the local Computer Store. You reach to the back of the Apple to turn it on and when you push the power switch to on, it has a strange feel like something is broken (no click or pop); the Apple power light doesn't come on and the disk drive doesn't spin.

Congratulations-you have just found the one major weak point in your Apple II Computer-the mechanical failure of the AC power switch in the power supply. If you have reached this point in the life of your Apple, you don't have very many options left open to you. Apple doesn't want you to get inside the power supply box and see just how cheaply it is made and therefore have riveted the case closed in each of the corners. At this time, the only thing you can do is to take your Apple to an Apple Dealer where he will be able to exchange the power supply under Apples exchange module policy. An exchange power supply module will cost you about $60.00 plus labor if only the switch is defective. Apple was really thinking and also sealed a fuse and a low value power resistor inside the power supply box and if these items are causing the Apple not to work, the exchange module price jumps to about $100.00 plus labor. Under any circumstances DO NOT DRILL OUT THE RIVETS or attempt to open the power supply box (or pry out the power switch) because any signs of tampering with the power supply will not allow the Dealer to exchange the module and he will have to charge you $300.00 for a new supply. Apple is very definite about how the power supply must be returned and in what condition so if you have opened the box, the Dealer will not accept the supply for an exchange. You should also note there are other things that can go wrong with a power supply other than the switch and the fuse. The most frequent thing that can happen is the switching transistor can short and die from voltage breakdown-this is very common in the older Apple power supplys. Another thing that can happen is the diodes in the supply can short or one of the supply voltages can become too high or low. In these last two cases, the best thing to do is bite the bullet and pay the $100.00 exchange price. Most older Apple power supplys usually develop other problems rather than the switch so you are better off making an exchange with the Apple Dealer.

In most power supplies, the switch will give you some warning of it's impending (and expensive) death such as having to flip the switch on and off a few times before the Apple comes on or by just the mechanical feel of the switch when it is turned on and off. The switch will not make a clicking noise and will feel very loose. It may even try and fall back to the off position. If your Apple power supply is at this point in it's life, RUN to the nearest hardware store and buy a multiple plug-in outlet strip with a switch such as Wards part number 84 A 60942 on page 682 of the Wards Spring and Summer 82 catalog or Radio Shack number 61-2619 on page 77 of their 1982 Catalog #341. You should now tape the switch in the Apple in the "ON" position and use the switch on the outlet strip to turn your system on and off. Another alternative is to buy some type of in line cord switch and put it in the Apple power cord. You might also buy one of the fans that has a built in power switch and run your Apple through it. I believe both of the mentioned outlet strips have replaceable outlets that can be modified to use an outlet that has a built in switch so you can control the Apple separately from the rest of your computer system. For those of you using a Zenith Video Monitor, you should be aware that the switch on the front of the set doesn't turn off the power to the monitor but just disconnects the DC power to the electronics inside. The power supply is on all the time the set is plugged into the wall outlet.

It might be too late for some of you but do remember to unplug your computer system from the house wiring system and

your modem from the telephone line when it is not in use-lightning can jump across a power switch with great ease and can kill an Apple. (I have just finished repairing an Apple III system that lightning got into-very very expensive.)

One other item worth mentioning is made by General Electric called a GE MOV-750. This device looks like an overgrown capacitor (about the size of a quarter) that is put across the 120 volt power line. This device will protect your Apple from voltage spikes caused by lightning or large power devices being turned on or off. GE also makes a device called a Voltage Spike Protector which sells for about $10.00. Radio Shack also sells this device-very cheap insurance for your Apple. I hope the preceding information might save you some money by not having to replace the power supply because of switch problems.

JDQ

FOR YOUR INFORMATION

Das Numberkruncher.....

There was a half page writeup, and a photograph, about the Basis computer. Look for it in the Apple Orchard, July-August 1982, page 31. It has some interesting possibilities.

The Pineapple.....

The next time you go to the grocery store, or whatever, look at the ad on page 102 of Radio - Electronics for August 1982. If anyone has factual information about this 'sort of familiar' looking device let us know.

The Computer Shopper.....

This is the 'yellow sheet' of the computer industry, and is well worth your looking at a copy. It's hard to find though, so you might drop them a line and ask for a sample copy and subscription information. Mail to: P. O. Box F, Titusville, FL 32780.

WHERE IS DOS?

by Louis D. Breyfogle

There are several good track and sector routines around which make it easy to modify DOS that is resident on a disk. Several good articles have been written about customizing a disk to suit ones fancy. Many of these articles make use of rather large and complex programs dedicated to one particular change. Such programs are fine, especially if you are receiving payment for each page you publish, but they do little to promote a basic understanding of how things work. It seems that one of the better kept secrets about APPLE DOS is the information that lets you cross reference a given track and sector on a disk with a particular RAM address in your Apple. The table given below will do just that for a STANDARD Version of APPLE'S 3.3 DOS in a 48K machine.

Disk TK,SC	RAM Addr HEX	DEC	Disk TK,SC	RAM Addr HEX	DEC
00,00	B600	46592	01,03	A400	41984
00,01	B700	46848	01,04	A500	42240
00,02	B800	47104	01,05	A600	42496
00,03	B900	47360	01,06	A700	42752
00,04	BA00	47616	01,07	A800	43008
00,05	BB00	47872	01,08	A900	43264
00,06	BC00	48128	01,09	AA00	43520
00,07	BD00	48384	01,0A	AB00	43776
00,08	BE00	48640	01,0B	AC00	44032
00,09	BF00	48896	01,0C	AD00	44288
00,0A	--	--	01,0D	AE00	44544
00,0B	--	--	01,0E	AF00	44800
00,0C	9D00	40192	01,0F	B000	45056
00,0D	9E00	40448	02,00	B100	45312
00,0E	9F00	40704	02,01	B200	45568
00,0F	A000	40960	02,02	B300	45824
01,00	A100	41216	02,03	B400	46080
01,01	A200	41472	02,04	B500	46336
01,02	A300	41728			

The two addresses left blank are the locations of the relocation code which is lost after DOS is relocated. Now, armed with the above information, and a track and sector routine, you can change DOS commands, or otherwise personalize your disks as you see fit.

LDB

TUTORIAL ON AN ELEMENTARY PASCAL PRINTER DRIVER

by Ron Abbott

After recieving my Epson Graftax roms, I decided that it would be a real treat to finally print out all those neat hi-res pictures, but the job of writing of the Pascal driver for the Epson, turned out to be more of a job than I had anticipated (like most jobs!).

This article is an elementary tutorial on how to map the hi-res memory page to a Epson Printer. By the way, this printer must have installed the Graftrax option. This is standard on the Mx-70, and Mx-100, but optional on the Mx-80 and Mx-80ft. The acompaning program is written in Pascal, but BASIC programmers will/should find most of this article of some use.

Before we talk about how to send data to the printer, lets talk about how the hi-res memory is organized (organized?) in the Apple. There are two hi-res pages, page 1 starts at decimal address 8192 and extends to address 16375. Page 2 starts at 16384 and extends to location 24576. We will deal with only page 1 in this article.

These memory locations contain the information needed to turn on certain dots on the hi-res screen. We will talk in terms of a high resolution screen that is 280 dots horizontal by 192 dots vertical. The Apple uses 1 bit to display 1 dot on the screen. One byte (8 bits) controls only 7 dots, the 8th bit in each byte controls color. So to display 280 dots takes 280 bits, 280/7 bits = 40, the number of bytes needed to control each horizontal line on the screen.

Refer to chart 4, this is reprinted from The Apple Orchard fall '80' issue, an it shows all the memory locations controlling the 192 rows. What a mess! At first glance there appears to be no logic to the way memory locations skip around. Starting with row 0 is memory location 8192, then each suceeding row's starting address is incremented by 1024, then when 8 rows have been mapped, increase the origin (8192) by 128 and start again adding 1024 to each succeeding row. After 64 rows, add 40 the the origin and start again for 8 blocks of 8 rows. All this jumping around may seem to be an unnecessary complication, but it is done this way to reduce screen flicker. While to trying to figure an easy way to compute indivual addresses, I remembered an article I had seen in Microcompuuting (Sep 79) by Darrell Smith concerning the physical layout of the memory. After consulting this article I came up with this formula.

Given any line number (n) 0<=n=<191 the starting row's address is calculated with the following:

Group = n divided by 64 { this tells you what group of 64 rows the address is in, see chart 4

Sub Group = (1 divided by 8) modulus 8 { this tells you what sub group of 8 rows the target lies within }

Line in Sub Group = n modulus 8

Hi-res Memory Location = 8192 + (Group * 40) + (Sub Group * 128) + (Line in Sub Group * 1024)

As an example lets find the starting address of line 142. We substitute into the expression thusly:

Group = 142 / 64 = 2

Sub Group = (142 / 8) modulus 8 = (17) mod 8 = 1

Line in Sub Group = (142) modulus 8 = 6

Hi-res Memory Address = 8192 + (2 * 40) + (1 * 128) + (6 * 1024) Hi-res Memory Address = 8192 + (80) + (128) + (6144) Hi-res Memory Address = 14544

If you again refer to chart 4 you can see that row 142 starts at memory address 14544 and extends to 14583. So now that we know how to compute the starting

address of any given line number 0-191, it should be a relatively simple matter to map the hi-res page to the Epson printer. Yes, it should be simple but there are considerations that make this more difficult than first imagined.

The Apple maps 1 byte of information as 0-7 dots in a horizontal plane, a dot matrix printer (like the Epson) takes 1 byte of information and translates this into 0-7 dots vertical. This explains why most hi-resolution pictures are printed at 90 degrees rotation from what appears on the t.v. or monitor. The Pascal program included will print hi-res page 1 image in the 60 dots per inch (horizontal) mode. If you intend to write a program that makes double sized images, then it must be printed with the x and y axis reversed. Why? Well the Apple's hi-res page is 280 X 192, if we double this it becomes 560 X 384. Since the Epson can only print 480 dots on any one horizontal line, it becomes ovious that we must rotate the picture so that the y axis (384) is printed horizontal.

By the way, since the Apple uses only 7 bits of 8 possible bits to trasmit characters, the Apple can only print line lengths of 0-127 and 256-383 dots (in the 60/in mode). There is documentation in the Graftax booklet on this limitation. The program that actually dumps the hi-res page to the printer is called Graphic Utility (properly called a unit in Pascal, when installed in the System.Library). What this program does, is to set up two control loops, controlled horizontally by values 0-39 and vertically by 0-191. The program starts by reading bytes from memory on what appears on your screen as the top right corner, downward from row 0 to row 191. Then it shifts on byte left, and repeats the process for 39 more times. In other words read column 39 (192 bytes) print it out as row 0, get column 38, print row 1 ect (see fig. 1).

By the way, BASIC programmers listen up, you can compute the actual address and value of any byte, given the x,y coordinates of the screen dot you want to turn on. Lets take for example the screen location 22,142. 22 is the x coordinate and 142 is the y coordinate. We already know how to compute the starting address of row 142 from the previous example now all we have to do is figure out what byte in row 142 controls 22,142 and what value that byte should have.

Here's how you go about it. The actual byte address is computed thusly; target address = row address + (x coordinate div 7), so in our example target address = 14544 + (22 / 7) = 14547. To determine what value should be poked/stored at memory location 14547 to turn on a dot at 22,142 you use this simple equation. Value to be stored = 2 ** (x mod 7) = 2**1 = 2. { the ** symbol means raised to the power of}

Summarizing this, to turn on a hi-res dot at screen location 22,142 poke/store the decimal value 2 at memory location 14547. These proceedures are already provided for you in BASIC as well as Pascal, so no need to dwell further.

BASIC programmers could write a graphic driver in BASIC following the algorithm present here and it would probably work just fine, except for one thing, it would be painfully slow. A better solution is to write it in assembly language. If you're not a machine language programmer, take heart, write it in BASIC and beg borrow or steal a APPLESOFT COMPILER to do the translation.

BASIC types, take a hike, Pascal people listen up! There are two programs included with this article. One is the graphic utility called oddly enough 'Graphic Utility'. This unit should/could be installed in your system library following the instructions on page 188 of your Apple Pascal Operation System Manual. These instructions are fairly good, so you shouldn't have any real problems installing this unit.

Once your have correctly installed this unit, you will have the ability under program control to save a hi-res picture to disk, recall a picture from disk to memory, and print this picture out. The three commands would look like this:

Save_hires_pic(name of pic), Load_hires_pic(name of pic), Epson_dump(inverse). Inverse is type boolean so set it to true or false before calling Epson_dump, name of pic is the name of the disk file that you supply.

The other program called 'Graphic_test' is a trivial example of how to use these proceedures. The program draws a 3-D square on the monitor, prompts for a name, files it, recalls it, and optionally dumps the image to your printer.

It is perhaps a worthwhile exercise to compare the printed image to the hi-res picture on the monitor. You will notice the the square is not square on the hard copy. This is because the Epson prints 60 dots/in horizontally, and 72 dots/in vetically. The aspect ratio is .83 (horizontal/vetical). For it to appear square would require a aspect ratio of 1.00. When printing in the 120 dots/in mode the ratio is 1.66. The only way to correct this is to buy a plotter or compensate for this effect with software.

By the way the hi-res screen dump to the printer is not exactly what I would call fast, time: 2 min 6 sec. A Applesoft version might take 5 min or more. If someone writes a Applesoft version and compiles it, I would be interested to know how faster than Pascal it might be. For comparisons, I timed a Silentype Printer and it came out to about the same 2 min 6 sec!

Note 1: The modulus function refered to in the text, looks like this 'x mod y'. What this means is divide x by y and give the remainder. Examples 3 mod 2 =1, 8 mod 3 = 2, 121 mod 121 = 0

Note 2: The above works well with either Graftrax or Graftrax+.

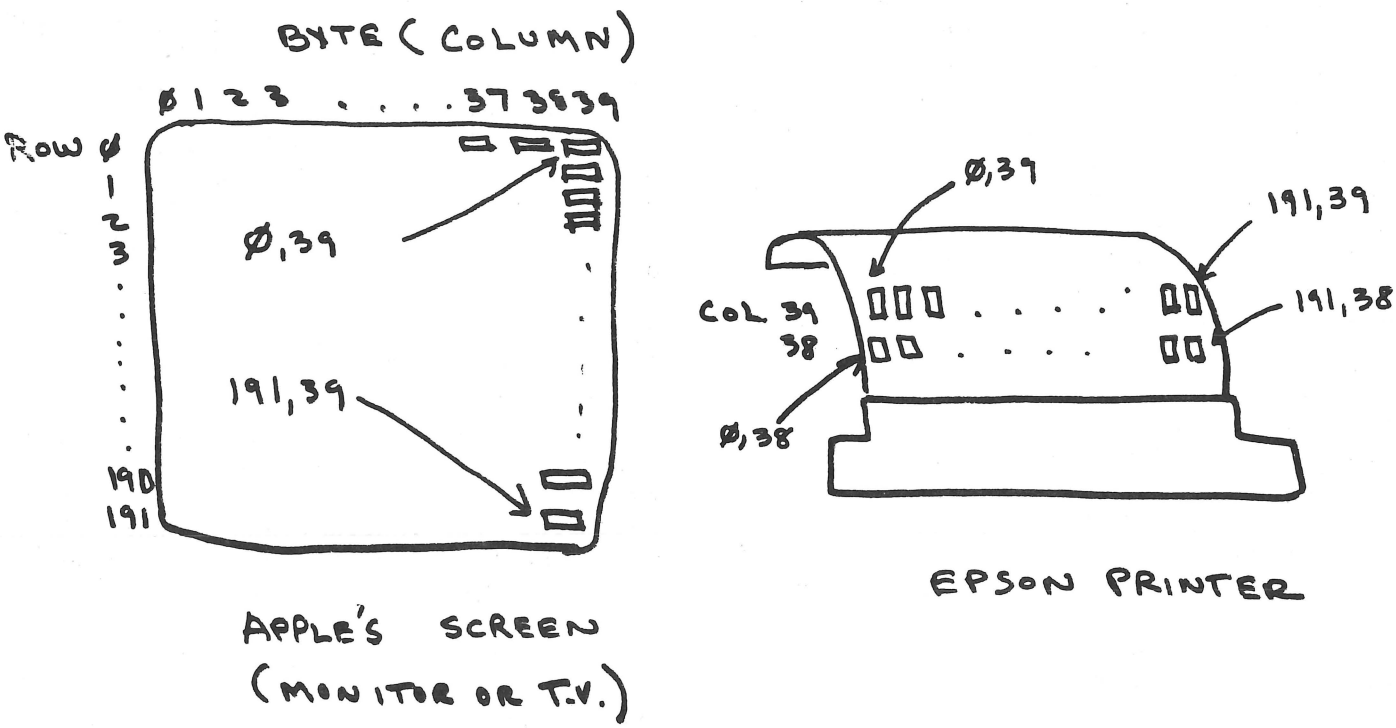

FIG. 1

0 8192- 8231	48 8960- 8999	96 8744- 8783	144 8528- 8567
1 9216- 9255	49 9984-10023	97 9768- 9807	145 9552- 9591
2 10240-10279	50 11008-11047	98 10792-10831	146 10576-10615
3 11264-11303	51 12032-12071	99 11816-11855	147 11600-11639
4 12288-12327	52 13056-13095	100 12840-12879	148 12624-12663
5 13312-13351	53 14080-14119	101 13864-13903	149 13648-13687
6 14336-14375	54 15104-15143	102 14888-14927	150 14672-14711
7 15360-15399	55 16128-16167	103 15912-15951	151 15696-15735
8 8320- 8359	56 9088- 9127	104 8872- 8911	152 8656- 8695
9 9344- 9383	57 10112-10151	105 9896- 9935	153 9680- 9719
10 10368-10407	58 11136-11175	106 10920-10959	154 10704-10743
11 11392-11431	59 12160-12199	107 11944-11983	155 11728-11767
12 12416-12455	60 13184-13223	108 12968-13007	156 12752-12791
13 13440-13479	61 14208-14247	109 13992-14031	157 13776-13815
14 14464-14503	62 15232-15271	110 15016-15055	158 14800-14839
15 15488-15527	63 16256-16295	111 16040-16079	159 15824-15863
16 8448- 8487	64 8232- 8271	112 9000- 9039	160 8784- 8823
17 9472- 9511	65 9256- 9295	113 10024-10063	161 9808- 9847
18 10496-10535	66 10280-10319	114 11048-11087	162 10832-10871
19 11520-11559	67 11304-11343	115 12072-12111	163 11856-11895
20 12544-12583	68 12328-12367	116 13096-13135	164 12880-12919
21 13568-13607	69 13352-13391	117 14120-14159	165 13904-13943
22 14592-14631	70 14376-14415	118 15144-15183	166 14928-14967
23 15616-15655	71 15400-15439	119 16168-16207	167 15952-15991
24 8576- 8615	72 8360- 8399	120 9128- 9167	168 8912- 8951
25 9600- 9639	73 9384- 9423	121 10152-10191	169 9936- 9975
26 10624-10663	74 10408-10447	122 11176-11215	170 10960-10999
27 11648-11687	75 11432-11471	123 12200-12239	171 11984-12023
28 12672-12711	76 12456-12495	124 13224-13263	172 13008-13047
29 13696-13735	77 13480-13519	125 14248-14287	173 14032-14071
30 14720-14759	78 14504-14543	126 15272-15311	174 15056-15095
31 15744-15783	79 15528-15567	127 16296-16335	175 16080-16119
32 8704- 8743	80 8488- 8527	128 8272- 8311	176 9040- 9079
33 9728- 9767	81 9512- 9551	129 9296- 9335	177 10064-10103
34 10752-10791	82 10536-10575	130 10320-10359	178 11088-11127
35 11776-11815	83 11560-11599	131 11344-11383	179 12112-12151
36 12800-12839	84 12584-12623	132 12368-12407	180 13136-13175
37 13824-13863	85 13608-13647	133 13392-13431	181 14160-14199
38 14848-14887	86 14632-14671	134 14416-14455	182 15184-15223
39 15872-15911	87 15656-15695	135 15440-15479	183 16208-16247
40 8832- 8871	88 8616- 8655	136 8400- 8439	184 9168- 9207
41 9856- 9895	89 9640- 9679	137 9424- 9463	185 10192-10231
42 10880-10919	90 10664-10703	138 10448-10487	186 11216-11255
43 11904-11943	91 11688-11727	139 11472-11511	187 12240-12279
44 12928-12967	92 12712-12751	140 12496-12535	188 13264-13303
45 13952-13991	93 13736-13775	141 13520-13559	189 14288-14327
46 14976-15015	94 14760-14799	142 14544-14583	190 15312-15351
47 16000-16039	95 15784-15823	143 15568-15607	191 16336-16375

CHART #4

This chart is a complete list of all 192 Hi-res lines as they appear on the screen. The two adjacent figures are the first and last bytes of RAM used to control that Hi-res line.

```
UNIT GRAPHIC_UTILITY;
INTRINSIC CODE 27 DATA 28;
INTERFACE
VAR IMAGE : FILE;
PROCEDURE SAVE_HIRES_PIC( FILE_NAME : STRING );
PROCEDURE LOAD_HIRES_PIC( FILE_NAME : STRING );
PROCEDURE EPSON_DUMP( INVERSE : BOOLEAN );


IMPLEMENTATION
Const
   page_1 = 8192;
Type
   data   = 0..255;

Var
   cheat : record case boolean of
           true : (intpart:integer);
           false: (ptrpart:^integer);
         end; {record}

Function Peek(address : integer):data;

Type
  byte = packed array[0..1] of 0..255;
Var
  mem_loc : record case boolean of
            true:(addrs:integer);
            false:(ptr:^byte);
          end; {record}
Begin
  mem_loc.addrs:= address;
  peek:=mem_loc.ptr^[0];
End; {peek}

Procedure error(result:integer);
Begin
    case result of
      1: writeln('Bad Block on diskette');
      2: writeln('Bad Device');
      3: writeln('Illeagal Operation { printer? }');
      4: writeln('Hardware barf');
      5: writeln('Lost Device {not on line}');
      6: writeln('Lost File');
      7: writeln('Bad title');
      8: writeln('No room for them at the inn');
      9: writeln('No device');
      10: writeln('No such File');
      11: writeln('Duplicate File');
      12: writeln('File already open');
      13: writeln('File already closed');
      14: writeln('Bad input format,expected real or integer');
      15: writeln('Data ring buffer overflow');
      16: writeln('Write protect error');
      64: writeln('Device error')
    end; { case result }
```

```
 end; { error }

Procedure Load_hires_pic ;
{ blockread hi-res buffer into memory at dec 8192 }

Var
  i,io : integer;

Begin
  cheat.intpart:=page_1;
    (*$I-*){ stop i/o checking to prevent crash on error }
  reset(image,file_name);
    if ioresult <> 0 then error(ioresult);
  io:=blockread(image,cheat.ptrpart^,16);
    if (ioresult<>0) or (io<>16) then error(ioresult);
  (*$I+*);
  close(image);
End; { Load_hires_pic }

Procedure Save_hires_pic;

Var
   i,io : integer;
Begin
   cheat.intpart:= Page_1;
   (*$i-*) {stop i/o checking to prevent crash on error}
   rewrite(image,file_name);
   io:=blockwrite(image,cheat.ptrpart^,16);
      if (ioresult<>0) or (io<>16) then error(ioresult);
   (*$i+*)
   close(image,lock);
      if (i<>0) or (io<>16) then error(i);
end; { save_hires_pic }

Procedure Epson_dump;
VAR
              null : data;
              bite : data;
                ch : char;
         dummy_data : integer;
      hires_mem_loc : integer;
       delay,column : integer;
        line_number : integer;
           lf,group : integer;
           subgroup : integer;
   line_in_subgroup : integer;
   data_to_printer  : packed array[0..255] of data;
     printer_setup  : packed array[0..3] of data;
                ff  : packed array[0..1] of data;
              crlf  : packed array[0..3] of data;
Begin
   null:=0;
   printer_setup[0]:=27;printer_setup[1]:=75;
   printer_setup[2]:=0; printer_setup[3]:=1;
   crlf[0]:=27;crlf[1]:=65;crlf[2]:=7;crlf[3]:=138;
   { 27=esc, 65=A, 7 is # dots vert, 138 is dec 10 (cr) }
```

```
   ff[0]:=0;ff[1]:=140;
     for column:=39 downto 0 do
        begin
           for line_number:=0 to 191 do
             begin
                group:=line_number div 64;
                sub_group:=(line_number div 8) mod 8;
                line_in_sub_group:=line_number mod 8;
                hires_mem_loc:= ( 8192 + column + ((group * 40) +
                                  (sub_group * 128 ) +
                                  (line_in_sub_group * 1024) ));
                bite:=peek(hires_mem_loc);

                  if not inverse then data_to_printer[line_number]:=bite + 128
                     { add 128 to value in hi-res memory because
                       Apple can't pass dec 10 and dec 16 correctly
                       to the printer, instead of sending values
                       0-127, now send 128-255 }

                  else data_to_printer[line_number]:=127 - bite + 128;
                     { this makes it inverse }

             end; { line_number }

        for dummy_data:=192 to 255 do
        data_to_printer[dummy_data]:=null;
          {now pad 64 null bytes to array to fill it up all
           256 bytes, Epson can accept 0-127 or 256-383 byte
           long graphic line (because Apple sends only 7 bits)}

        unitwrite(6,printer_setup,4);
        { tell the printer to expect 256 bytes of graphic data
          to be used in the 60 dot/in bit graphics mode.
          (send esc,"K",0,1 to printer)}

        unitwrite(6,data_to_printer,256);
         { now send the 256 bytes, 192 of real data and 64 null bytes }

        unitwrite(6,crlf,4);
        { now do a 7 dot vertical line feed }

     End; { for column }
        { at this point a hi-res picture should be printed on the MX-80 }
   unitclear(6);
   unitwrite(6,ff[1],1);
end; { Epson_dump }

BEGIN END.
```

```
PROGRAM Graphic_test;
uses GRAPHIC_UTILITY,
     TURTLEGRAPHICS;
var ch : char;
    album : string;
    inverse : boolean;
    ans   : char;
Begin
   initturtle;pencolor(none);moveto(70,168);
   wstring('This is a 3-D square?');
   moveto(91,91);pencolor(white);
   moveto(119,63); moveto(175,63);
   moveto(147,91); moveto(175,63);
   moveto(175,119); moveto(147,147);
   viewport(91,147,91,147);
   fillscreen(white);
   read(ch);{ hold in grafmode until user ready }
      textmode;
      writeln(chr(26)); { change to page(output) for apple }
      write('Name to file screen image => ');readln(album);
      save_hires_pic(album);
      writeln('Image saved on disk as ',album);
      writeln('Press any key, hi-res screen will be cleared');
      writeln('and the file ',album,' will be loaded from disk');
      read(ch); fillscreen(black);
      load_hires_pic(album);
      write('Do you want image printed on Epson? ( Y or N )');
      readln(ans);
        if (ans='N') or (ans='n') then exit(program);
      write('Want image inverse? ( Y or N )');
        readln( ans);
          case ans of
             'Y','y': inverse:=true;
             'N','n': inverse:=false;
          end;{case}
      grafmode;
      epson_dump(inverse);
      textmode;
         writeln('Image should be on printer now');
         write('press any key');read(ch);
end.
```

SEED UNCLASSIFIED ADS

PAPER-9½ X 11" PIN-FEED-CONTINUOUS, 1 PART,1250 SHEETS, WHITE 20# BOND--- $22
Call Frank 499-2192

§»»»